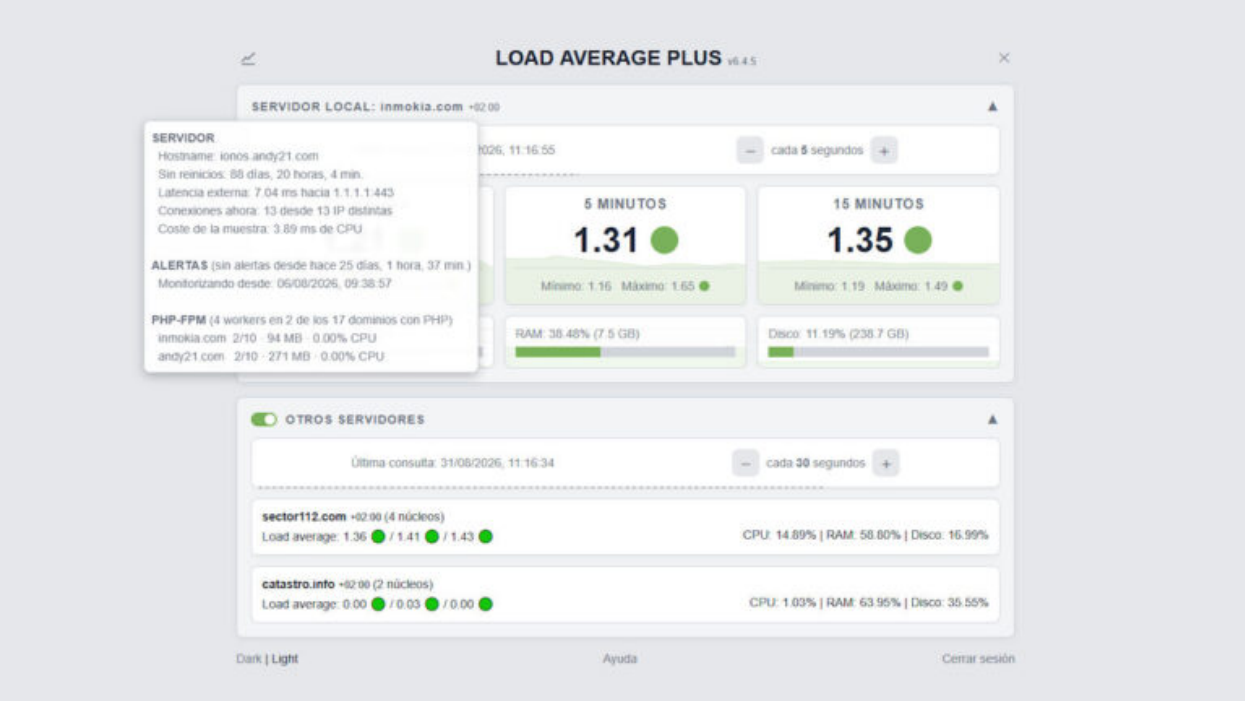
Tu servidor web no está siendo atacado



Cada vez que una web carga lenta aparece la misma palabra: ataque. Pero tu servidor casi nunca está siendo atacado, y confundir las dos cosas hace perder días buscando en el sitio equivocado.

Servidor hackeado o servidor lento

Conviene separarlas antes de seguir, porque tienen poco que ver entre sí.

Si el contenido de la web ha cambiado sin que nadie lo tocara, si aparecen páginas que no escribiste o directamente un mensaje de alguien reclamando la autoría, eso sí es una intrusión. Alguien ha entrado. Es un tema serio, con su propio protocolo, y no va de esto.

Lo otro es distinto: la web sigue siendo la tuya, nadie ha entrado,

pero carga lenta o se cae a ratos. Ahí no hay intrusión, hay consumo. El servidor está trabajando más de lo que puede, y quien lo provoca casi nunca ha entrado en ninguna parte.

Por qué se ralentiza un servidor web

La lista es larga y casi ninguna entrada es un ataque.

- **Rastreos masivos.** Buscadores y, cada vez más, robots que recopilan contenido para entrenar modelos de inteligencia artificial. Hacen su trabajo, pero a un ritmo que ninguna web pequeña está preparada para absorber.
- **Robots que mienten.** Se identifican como Google o Bing sin serlo. Se detectan porque su dirección IP no corresponde con quien dicen ser.
- **Caché que se regenera.** Cuando se vacía, o cuando una página no se puede cachear, cada visita se construye desde cero. Es uno de los factores que más pesa en cuánto tiempo tarda en cargar una web. Las páginas de búsqueda y las de carrito están siempre en ese grupo.
- **Tareas programadas pesadas.** Copias de seguridad, antivirus, sincronizaciones con un ERP, importaciones de catálogo. Suelen ir de madrugada, y si coinciden con tráfico real se suman a él.
- **Tareas que se solapan consigo mismas.** Se lanzan cada cinco minutos, tardan siete, y acaban ejecutándose tres a la vez. El servidor va acumulando copias del mismo trabajo hasta ahogarse.
- **Llamadas a servicios externos.** Una página que por dentro consulta una API ajena deja el proceso esperando mientras el visitante ve la rueda girar. Con veinte visitas simultáneas, veinte procesos parados sin hacer nada.
- **Consultas lentas por falta de índices.** El caso clásico: funcionó durante años y de pronto deja de funcionar, porque una tabla ha crecido lo suficiente para que una consulta mal planteada pase de instantánea a costosa.
- **El vecino ruidoso.** En un servidor virtual compartes hierro con otros clientes. Si uno se desmadra, lo notas sin tener culpa ni control. Es lo único de esta lista que no puedes arreglar.
- **Correo saliente atascado.** Una cola de envíos puede consumir

tanto como el tráfico web, y no aparece en ningún registro de acceso.
- **El propio código.** La que menos gusta reconocer. Una consulta dentro de un bucle, un plugin que hace más de lo que parece, o una línea puesta con buena intención que resulta tener un efecto que nadie previó.

De esta lista, las tres últimas semanas me han tocado cuatro.

Cuatro días buscando dónde no era

En una tienda online que administro, con más de 1.500 productos y unas 3.500 entradas reales en base de datos contando variaciones, la web empezó a ir lenta de vez en cuando. Nada dramático: lenta, luego normal, luego lenta otra vez. Sin patrón.

Miré por encima con las herramientas de siempre, esas de mirar Apache en tiempo real desde la consola, y no saqué nada en claro. Sospeché del hosting. Después de un cambio de configuración de PHP que había hecho días antes, y que revertí sin que mejorara. Después incluso eché mano de Load Average Plus, mi propio monitor de servidores, que acababa de actualizar y me marcaba la CPU al 100% de forma sostenida.

Ninguna de las tres era la causa. Y la CPU al 100% resultó ser un dato falso, sobre eso vuelvo al final.

La respuesta estaba en los registros de acceso, y las cifras no dejaban lugar a dudas:

```
18 al 24 de agosto:   1.386 a 5.785 peticiones diarias
25 de agosto:       135.720
26 de agosto:        60.640
27 de agosto:       175.507
```

El agente era meta-externalagent, el robot que Meta usa para entrenar sus modelos de inteligencia artificial. Y el 99,7% de esas peticiones iban a un solo sitio: las páginas de resultados de una búsqueda interna concreta, paginadas hasta donde llegara.

Ese es el peor sitio posible por donde entrar. Las páginas de búsqueda no se cachean nunca, porque son dinámicas por definición. Consultan la base de datos de forma pesada. Y paginar hasta la

número treinta obliga a recorrer y descartar cientos de filas para devolver veinte.

A partir de ahí se encadenó todo. Los procesos de PHP se ocupaban, Apache se cansaba de esperar y los mataba a los treinta segundos, y arrancar uno nuevo cuesta cargar WordPress entero, que con cada versión pide algo más al servidor. El servidor gastaba más en reponer procesos que en servir páginas.

La causa real: errores 404 que no lo eran

Aquí viene la parte incómoda, y la que de verdad importa.

Al revisar qué hacía el sitio con las direcciones que no existen, apareció esto: cualquier URL inventada devolvía código 200 y casi 600 KB de contenido. No un error. Una página completa.

```
/esto-no-existe-de-verdad/            200  ·  596.600 bytes
/producto/algo/inventado/xyz/         200  ·  596.795 bytes
```

Un rastreador usa el código 404 para saber que ha llegado al final. Sin él, cualquier dirección inventada parece contenido legítimo, así que puede generar rutas indefinidamente y todas responden. El espacio de direcciones deja de ser grande y pasa a ser infinito.

Meta encontró ese agujero y lo explotó a escala. Pero no lo creó. De hecho, el primero en dar con la URL fue otro rastreador, días antes, desde direcciones chinas y con agentes falsificados.

El robot fue el síntoma. La causa era una puerta abierta de par en par.

Cómo se arregló, con cifras antes y después

Tres capas, en este orden de importancia.

Que las direcciones inexistentes vuelvan a devolver error. Eso cierra el espacio infinito para cualquier robot, respete o no las normas. La página de sugerencias que ve el visitante sigue

funcionando exactamente igual: el código HTTP solo lo leen las máquinas.

Cerrar las páginas de búsqueda al rastreo en el robots.txt, que además es lo recomendable porque cuentan como contenido duplicado.

Y bloquear al agente concreto que se había desmadrado, comprobando antes que no era el mismo que sirve a las campañas publicitarias. De cada 859 peticiones que hacía Meta, 858 eran para entrenar su IA y una para la publicidad.

El resultado, medido:

```
Tiempo diario con la CPU por encima del 75%
  28 de agosto:  11:36:40
  29 de agosto:  00:00:01

Alertas y prealertas
  14 al 24 de agosto:  0 y 0
  25 al 28 de agosto:  4 y 9
  29 en adelante:      0 y 0
```

De once horas y media a un segundo. Esa es la línea que resume cuatro días.

Tres hallazgos que llevaban meses ahí

Cuando uno se pone a mirar de verdad, aparecen cosas que llevaban tiempo ahí.

La caché de código PHP se estaba quedando sin memoria y se vaciaba entera por su cuenta, dos veces en pocos días. Cuando eso pasa, todo el código se recompila justo cuando el servidor está más cargado. Subir su memoria de 128 a 256 MB llevó el acierto del 87% al 96%.

Otro servidor, mucho más pequeño, lee 36 GB de disco al día y pasa entre cinco y seis minutos diarios con el disco como cuello de botella, con una regularidad de reloj. Eso no es tráfico, es un proceso programado. Sigue sin resolverse.

Y la tercera es la que más me enseñó. Aquella CPU al 100% que me hizo sospechar de mi propia herramienta era un dato falso: tras reiniciar el servidor, los contadores del sistema vuelven a cero y la referencia guardada quedaba por encima de la lectura actual, así

que el último valor medido antes del reinicio se repetía indefinidamente. Medir mal es peor que no medir, porque no solo no ayuda: desvía la investigación.

Seguimos en las trincheras

No está todo resuelto. El disco de ese otro servidor sigue haciendo lo suyo cada día. Hay procesos que consumen más memoria de la que deberían. Y mañana aparecerá algo nuevo, porque siempre aparece.

La diferencia es otra. Antes, una web lenta era una sospecha y unos cuantos días de dar palos de ciego. Ahora hay datos: cuánto tiempo estuvo cada recurso por encima de su umbral, qué dominio concentraba los procesos, cuántas conexiones había y desde cuántas direcciones distintas. Con eso, un diagnóstico que costaba días cuesta una hora.

Esos datos los da Load Average Plus, la herramienta que desarrollé para la ocasión y que ya presenté como monitor de carga para servidores web. L.A.P. no evita los problemas: los hace visibles. Y lo que se puede medir, se puede mejorar.

Si te ha gustado este PDF, compártelo. Si no te ha gustado, díselo a Andy.